INFRASTRUCTURE OPTIMIZATION IN KUBERNETES CLUSTER

Darshan Haragi L[1], Mahith S[2], Prof. Sahana B[3]

[1]Student, Dept of Electronics and Communication, R.V. College of Engineering, Bangalore, INDIA -560059

[2]Student, Dept of Electronics and Communication, R.V. College of Engineering, Bangalore, INDIA -560059

[3]Assistant Professor, Dept of Electronics and Communication, R.V. College of Engineering, Bangalore, INDIA -560059

[1]darshanharagil.ec16@rvce.edu.in,[2]mahiths.ec15@rvce.edu.in

[3]sahanab@rvce.edu.in

Abstract:*Kubernetes is a compact, extensible, open-source stage for overseeing containerized responsibilities and administrations, that works with both decisive setup and robotization. Kubernetes is like VMs, however having loosened up isolation properties to share the Operating System (OS) among the applications. The container conversely with VM, has its own document framework, portion of Central Processing Unit(CPU), memory, process space and much more. Kubernetes cluster is a bunch of node machines for running containerized applications. Each cluster contains a control plane and at least one node. Infrastructure Optimization is the process ofanalyzing and arranging the portion of cloud resourceswhich power applications and workloads to augment the presentation and limit squander due to over-provisioning. In the paper, a "Movie Review System" web application is designed using GoLang for backend components and HTML, CSS and JS for frontend components. Using AWS, an EC2 instance is created and the web application is deployed onto EC2 and hosted in instance server. The web application is also deployed on Kubernetes locally using MiniKube tool. A performance analysis is performed for both the deployments on considering common performance metrics for both AWS EC2 / Virtual Machine (VM) and Kubernetes.*

Keywords: Kubernetes(k8s), Kubernetes Cluster, Virtual Machines (VM), Amazon EC2, MiniKube, Web Application.

1. INTRODUCTION

Kubernetes is a compact, extensible, open-source stage for overseeing containerized responsibilities and administrations, that works with both decisive setup and robotization.A virtual machine, is essentially a full machine running all the components, including its own Operating System, on top of the virtualized equipment. In the year 2014 an advancement the Kubernetes which is like VMs, however having loosened up isolation properties to share the Operating System(OS) among the applications was delivered. Kubernetes initially built by Google, at present is maintained by the Cloud Native Computing Foundation(CNCF). Kubernetes furnishes with the structure to run dispersed frameworks versatilely, considering the scaling and fail over of the application, giving deployment patterns and much more. Kubernetes gives the load balancing and storage orchestration, programmed binpacking, computerized rollouts and rollbacks and other network advancement related services. The Figure 1, shows the design of Kubernetes comprising of two fundamental parts: Master and Nodes. The master is divided into nodes which comprise of Pod, Container, Kubelet and Proxy.Kubernetes cluster is a bunch of node machines for running containerized applications. Each cluster contains a control plane and at least one node.

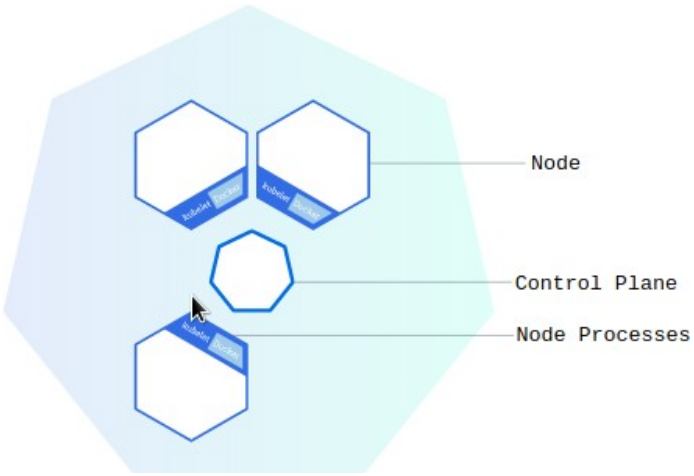

Figure 1: Kubernetes Cluster

The control plane keeps up the desired condition of cluster (like which applications are running) and the nodes run the applications and workloads. When the kubernetes is running it implies that the we are running the cluster. Each cluster comprises of individual 'Namespace' which permits to deal with multiple clusters in a similar physical cluster. Performance Optimization in the kubernetes is a significant worry recently, as kubernetes is highly extensible, open-sourceplatform for coordinating containerized workloads in cloud environment.

2. Literature Review

In research paper [1] the creators Nguyen Dinh Nguyen and Taehong Kim portray the Kubernetes design and the pioneer based system for keeping up predictable information stockpiling among reproductions of a stateful application in the Kubernetes group. The creators propose a pioneer political race calculation that not just works with the utilization of pioneer political decision in Kubernetes yet in addition equitably disperses the pioneers all through every one of the hubs in the group. The assessment results showed that the proposed BLD calculation can adequately adjust the quantity of pioneers among all hubs in the cluster. The viability of the BLD calculation was demonstrated through an exhibition assessment with various applications, showing that the throughput can be altogether improved by disseminating the quantity of pioneers equitably all through the hubs.

In research [2] by Ionut-Catalin Donca and group, the proposed arrangement was set off by the requirement for measure computerization, setup, execution of programming and the assembly of frameworks. Robotization of frameworks diminishes the expense of developing rehash capable science measure foundations and adds to the reproducibility of examination contemplates. The whole logical work process life cycle was rearranged by a framework introduced: programming setup, establishment and work process the board. The best technique for the inevitable arrangement of this framework is to handle the Autoscaler Cluster, which scales the cases dependent on various boundaries (CPU usage, memory use).

The creators Mulugeta Ayalew Tamiru, Johan Tordsson, Erik Elmroth and group in paper [3], express that 'Kubernetes has arisen as the accepted compartment arrangement stage in the cloud'. In the paper the creators presume that default design (CA) the autoscaler arrangements hubs at the hour of scale-out from just a single hub pool, while when arranged with hub auto-provisioning (CA-NAP) it arrangements hubs from numerous hub pools. We look at these two designs utilizing SPEC's auto-scaling execution measurements and financial expense of running the clusters.the effect of various arrangements and applications on the autoscaling execution and cost of running of a Kubernetes bunch. As the Kubernetes group autoscaler is profoundly configurable, it is fascinating to contemplate the effect of tuning extra boundaries on auto-scaling execution and cost saving.

In the paper [4] by Maria A. Rodriguez and Rajkumar Buyya, the creators have proposed the incorporated utilization of schedulers, autoscalers, and reschedulers as a component to make compartment coordination frameworks cloud-mindful. In this, the scheduler enhances the underlying arrangement of holders, the autoscaler empowers the current interest for assets to be met and underutilized or inactive hubs to be closure improve the framework's use and consequently lessen cost, lastly the rescheduler takes into account the underlying position of compartments to be updated at runtime to diminish discontinuity and combine burdens to energize better asset usage. Different rescheduling and auto-scaling system were proposed, carried out, and assessed.

Inpaper [5] by Junzo Watada, Arunava Roy and team, containerization provides many promising features like super lightweight, faster spin-up/down, efficient energy and resource utilization, impressive workload distribution capabilities, achieving server con- solidation, and many more, but at the same time it has few major problems such as weaker isolation, higher chance of container sprawl, lack of capable tools for container orchestration and cross-platform supports and container portability limitations. Uniker- nels provide VM-like isolation with significantly small footprints along with extremely fast booting. the recent technological developments in both VMs and lightweight virtu- alization, the increasing industrial acceptance of containerization will soon be challenged as the need of the day has shifted to the advantages.

In research [6] by Emiliano Casalicchio, propose and assess the presentation of KHPAA an improved variant of the KHPA auto-scaling calculation. The proposed arrangement influence, in a straightforward way, total utilization estimates rather then family member. KHPAA can be connected any current framework coordinated with Kubernetes without the need to change the framework arrangement. The presentation examination shows that the utilization of total measurements permits to appropriately control the application reaction time and to keep it beneath edges presented by administration level targets.

In paper [7] by Isam Mashhour Al Jawarneh and group, have examined compartment orchestration structure contrasting highlights and administrations offered by holder orchestrators. We have likewise planned a strong arrangement of measurements to think about their exhibitions at scheduling and administration the executives layers and we accept those measurements are reusable for a wide range of holder orchestrators. At last, we picked four agent holder orchestrators, specifically, Docker Swarm, Kubernetes, and Apache Mesos.

The authors Victor Medel, Omer Rana and team in paper [8], have outlined performance model for Kubernetes based deployment. The authors use a benchmark-based approach to better characterize behavior of a Kubernetes system (using Containers). Theproposedreferencemodelcanbeusedasupportfor:capacityplanningandresource management is outlined and applicationdesign.

3. Architectures and Models

3.1 Architecture of Web Application

Web application is a framework which outlines the key internal and external components, the cooperation of components like UIs, databases, CSV files and middleware systems. The web application architecture approaches and associates dissimilar components to improve web insight. The web application architecture comprises of segments which are categorized into two primary classes: user interface app and structural components. The Figure 2depicts the fundamental architecture of a web application.

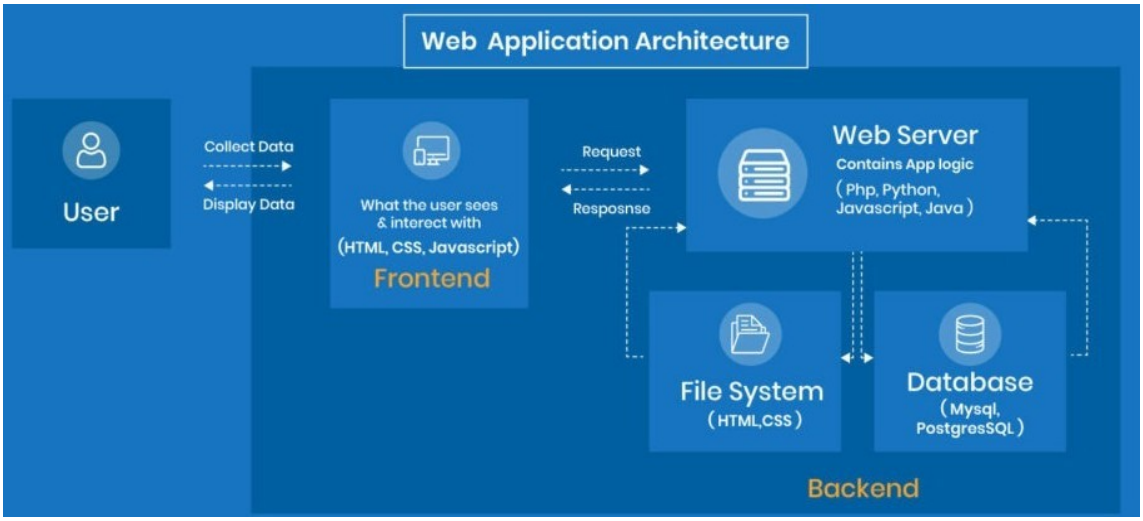

Figure 2: Basic Architecture of Web Application

User Interface Components

UserInterfaceComponentsarerelatedtosettings,configurationsanddisplayofweb applications. These components are responsible for creating and improving the experience and interfaces of the web application. These contain a number of components for instance, dashboards, configuration settings, statistical data, layouts,etc.

Structural Components

The structural components are responsible for fabricating the performance of web application. These components allow users to coordinate with the application. The structural components include:

The Client or Web Browser

The users can collaborate and communicate with web applications using this platform. HTML,CSS,andJavaScriptaretheprogramminglanguagesthatarevaluabletoshapethis component.

CSV Files or Database

The CSV offers relevant data or business logic that is overseen and stored by web application server. The database stores, retrieves, and delivers the data.

Web Application Server

Thewebservermanagesthefundamentalhubthatupholdsmulti-layerapplications andbusinesslogic.TheserverisusuallybuiltusingPHP,GoLang,Python,.NET,Java and Node.js. In the project the web server is designed using the 'GoLanguage'.

3.2 Kubernetes Architecture

Kubernetes follows the customer worker design where we have ace introduced on one machine and the hub on discrete Linux machines.It uses a master to manage Docker containers across multiple Kubernetes nodes. A master and its controlled nodes constituteaKubernetescluster.Theusercandeployanapplicationindockercontainers via the Kubernetes master. The Figure 3 is the Kubernetes architect consisting on Master and Workernodes.

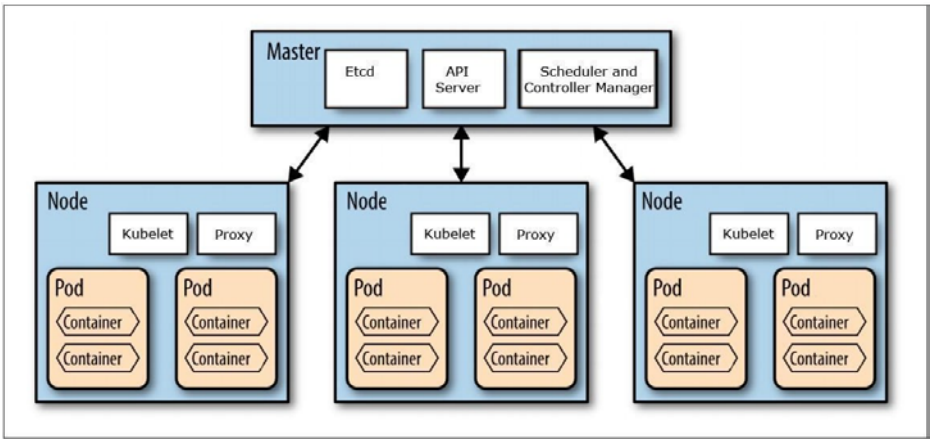

Figure 3: Kubernetes Architecture

Kubernetes Master Components

Kubernetes master is responsible for managing the cluster, coordinating activities inside cluster and communication with worker nodes to keep the application and Kubernetes running. The Master node is the entry point of all administrative tasks. The components of Kubernetes Master:

API Server

The users can interact and communicate with web applications via this platform. HTML,CSS,andJavaScriptaretheprogramminglanguagesthatarevaluabletobuildthis component.Programming Interface (API)serveristhesectionpointforalltheREST components usedtocontrolthecluster. All the regulatory tasks are carried outby Programming Interface (API)server within the master node. To make, erase, update or show in Kubernetes object it needs to go through this Programming Interface (API) server.Programming Interface (API) server approves and configures the Programming Interface (API) objects like ports, administrations, replication, regulators and arrangements and it is responsible for exposing Programming Interface for every operation, 'kubectl' is utilized to communicate with APItools. (API)

Scheduler

Scheduler is a service responsible for distributing the workload. Scheduler is liable for tracking the work load utilized at each worker node and placing the workload on which the resources are accessible. The scheduler is responsible for scheduling pods acrossavailablenodesdependingontherequirements referencedinconfigurationdocument.The scheduler is liable for workload utilization and allocating pod to a newnode.

Controller Manager

Controller Manager, also know as controllers, it is a daemon that runs in non- terminating loop and collects and sends the information to API server. The key regulators are replication regulator, endpoint regulator, namespace regulator, and administration account regulator.

etcd

etcd is a distributed key-value lightweight database. In Kubernetes, it is a central databaseforstoringthecurrentclusterstateatanypointoftimeandalsousedtostore the configuration details such as subnets, etc. The etcd is written inGoLang.

Kubernetes Worker Components

Kubernetes Workercontainsallnecessaryservicestomanagethenetworkingbetween containers,communicationwiththemasterandassignresourcestoscheduledcontainers. The components of Kubernetes Worker:

Kublet

Kublet is a primary node that communicates with the master and executes worker nodes inside the cluster. Kublet gets the pod specifications through the Application ProgrammingInterface(API)serverandexecutethecontainerassociatedwithpodsand ensurethecontainersarerunningandhealthy.

Kube-Proxy

Kube Proxy is the core networking component inside the kubernetes cluster. Kube Proxyisresponsibleformaintainingtheentirenetworkconfiguration.Kube-Proxymaintainsthedistributednetworkacrossallthenodes, podsandthecontainersandalsoexposes theservicesacrossoutsideworld.Kube-Proxyactsasanetworkproxyandloadbalancer foraserviceonasingleworkernodeandmanagesthenetworkroutingforTransmission Control Protocol(TCP) and User Datagram Protocol(UDP)packets.

Pods

Pod is a gathering of containers that are deployed together on a similar host.The pods can deploy multiple dependent containers together so it acts as a wrapper around these containers so we can interact and manage the containers primarily through pods.

4. Methodology

The web application is the integral part of the project. 'Movie Review System', a web application based on reviewing is designed and built. The web application consists ofaLoginpage,SignUppage,HomepageandaReviewpage. Auserisfirstdirectedto the Login page where he/she can directly login to their account using the user specific username and password if he/she has already created an account else the user can click on the SignUp button on the top left corner of the Login page and create an account. On successful creation of account and signing to the account, the user gets a setofmoviestochoose from,forwhichhe/shehastoreviewandtypecommentsandclickon Submit. After submitting review, the user needs to scroll to the bottom of the page and LogOut of his account, which he/sheis redirected to the LogInpage. TheFigure4isaflowchartshowingthecomplete methodology.

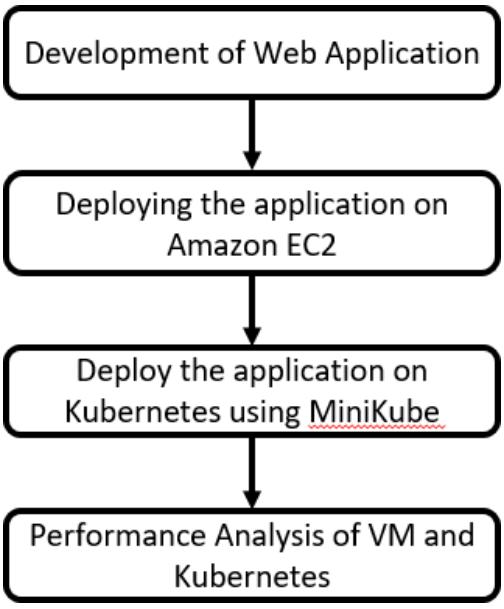

Figure 4: Complete Methodology

4.1 Deployment of Web Application on Amazon EC2

To launch a web application on Amazon EC2 instance, user needs to first create an accountinAmazonWebServicesinordertoSignIntoAWSManagementConsole.The user needs to perform followingsteps:

1. Choose EC2 from AWS Console and select Amazon Linux 2 Amazon MachineImage(AMI) as operatingsystem.
2. Selecting the instance type: 't2.micro'(Free tier) from the available set ofinstances.
3. Assigning a Public IP address, adding Storage, Configuring Security groupswhich are unique to eachinstance.
4. Onassigningalltheabovementionedcomponents,Launchingtheinstance.
   5.Creating an Keypair which is unique to eachinstance.
5. Connecting to the instance, installing and connecting to the Apache webserver.
6. After getting connected to the server, launching the web application on the AWS server.

4.2 Deployment of Web Application on Kubernetes

To deploy a web application on Kubernetes locally, user needs to use MiniKube and Katacoda. Katacoda is used as it provides free and in-browser Kubernetesenvironment. While deploying the web application on kubernetes the user needs to follow thesteps:

1. Creating a MiniKube Cluster (locally): by opening Kubernetes dashboard in the browser in minikube. Then a Katacoda environment is selected the port as30000.
2. CreatingaDeployment:usingKubectl,creatingdeploymentthatmanagesthepods and checking and setting the kubectl configurations. Deployment of application checks the health of Pod and restarts the Pod's if they get terminated whenbeing used.
3. Creating a Service: Initially the Pod is accessible only by its internal IP address, so Kubernetes service is performed to make the Pod accessible from outside the kubernetes virtual network.The application codes only listens to client requests from 'TCP port 8080'. The Katacoda environment automatically allots a 5-digit unique port number for every kubernetes service.
4. Oncompletionofabovementionedtasks,theapplicationopensonbrowserandits response, which isnoted.

5. Results and Discussion

Performance Analysis

The web application is deployed on Amazon EC2, a Cloud Compute Service provided by Amazon Web Services (AWS) and on Kubernetes using MiniKube. On hosting the application in EC2 server and locally on Kubernetes, the performance analysis is carried out using the Prometheus and Grafanna tools. The performance metrics considered are: CPU Utilization, Memory Footprints, Network Traffic, Time Synchronization and Entropy.
The Figures 5.1 depicts the basic CPU utilization graphs and the Figure 5.2 shows the exact usage percentage of CPU by user, system and when at idle state.

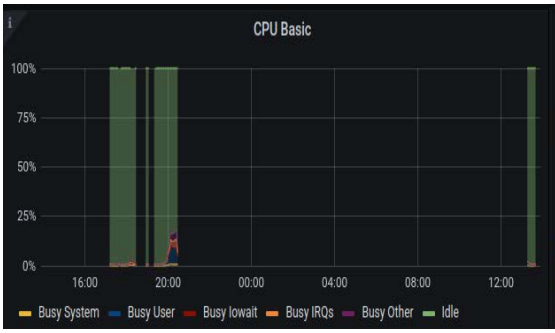

Figure 5.1: Basic CPU Utilization

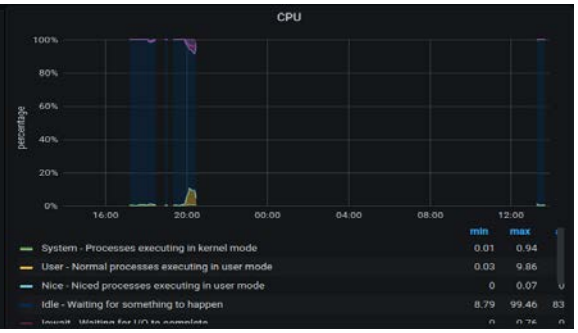

Figure 5.2: Detailed CPU Usage

The Figure 5.3 shows the exact memory consumed when the user accesses the application and the Figure 5.4 shows the memory consumed when user switched between pages.

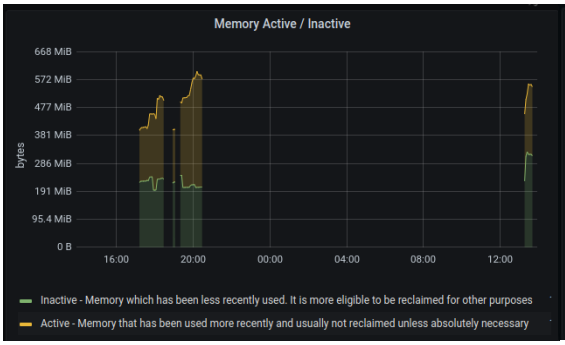

Figure 5.3: Memory Usage Status

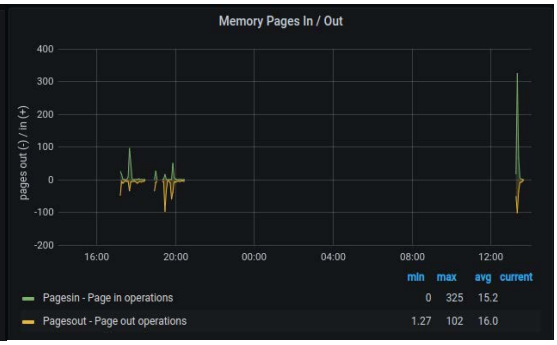

Figure 5.4: Memory used for Pages Accessed

The Figures 5.5 and 5.6 shows the time synchronized/elapsed between one LogIn and LogOut by the user to his/her account in the web application.

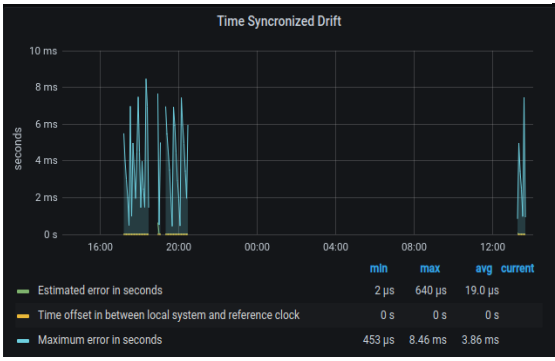

Figure 5.5: Time Synchronization

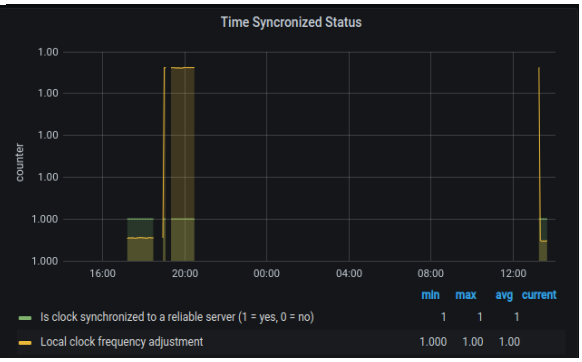

Figure 5.6: Time Usage Status

The Figure 5.7 depicts the Entropy of the web application i.e, the number of remote users accessing the web application at a given time.

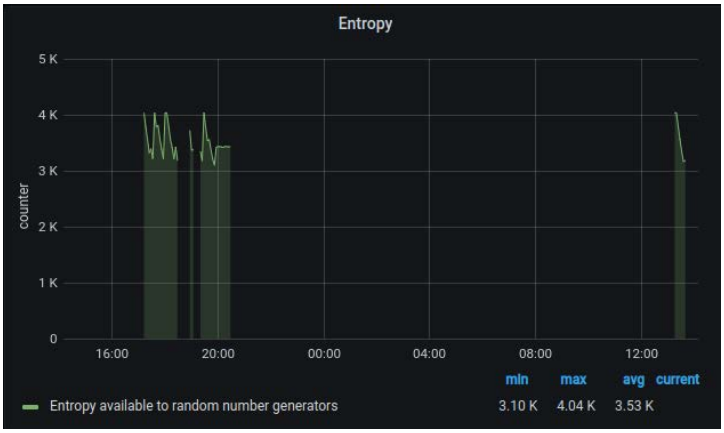

Figure 5.7: Entropy of web application

The Figure 5.8 shows Network traffic i.e, the amount of data which is moving across the web application/computer network. The Network traffic is analyzed in terms of data packets.

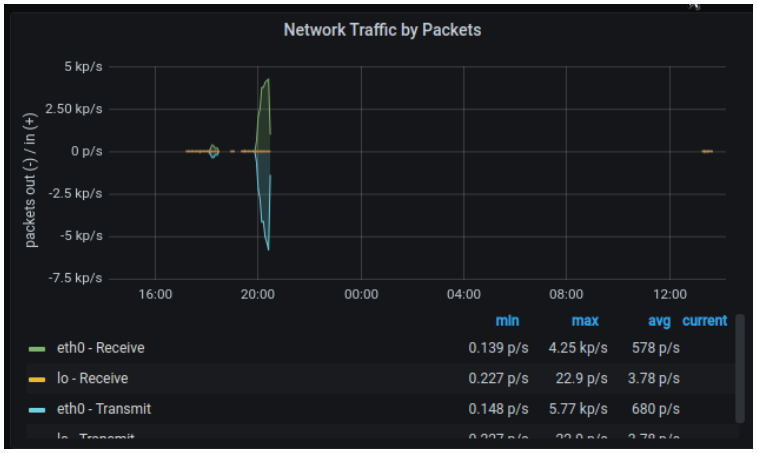

Figure 5.8: Network Traffic in terms of Data Packets

6. Conclusion

A 'Movie Review System', a web application based reviewing is designed and built. TheapplicationusesGoLangforbackendandHTML,CSSandJSforfrontend.Theweb applicationconsistsofaLoginpage,SignUppage,HomepageandaReviewpage.Auser has to LogIn to his/her account, select a movie, review and submit and then LogOutof his/heraccount.

Amazon Web Services, an open-source platform for a variety of cloud computing operations is used to host the web application on the server. The web application is deployed on the Amazon EC2 instance(Virtual Machine) and the instance is hosted on Amazon server. Using Minikube, the web application is deployed on Kuberneteslocally. The performance metrics considered in common are noted for both the VirtualMachine and Kubernetesdeployments.

The performance analysis is performed to conclude which among the Virtual Machine and Kubernetes performs better. The performance analysis is carried out considering the

performance metrics in common to both the VM and Kubernetes. Based on the results obtained after performance analysis, we conclude the Kubernetes performs better in terms of CPU utilization, Network traffic and Memory usage. Hence, from this study we conclude that the Kubernetes is better when compared to regular Virtual Machine when it comes to Load Balancing the Cloud Environments.

7. References

1. N.D.NguyenandT.Kim,"Balancedleaderdistributionalgorithminkubernetes clusters,"*Sensors*,vol.21,no.3,2021,ISSN:1424-8220.DOI:10.3390/s21030869. [Online].Available:https://www.mdpi.com/1424-8220/21/3/869.Sahoo, Doyen Liu, Chenghao Hoi, Steven. (2017). Malicious URL Detection using Machine Learning: A Survey.

2. C.Donca,C.Corches,O.Stan,andL.Miclea,"Autoscaledrabbitmqkubernetes clusteronsingle-boardcomputers,"in*2020IEEEInternationalConferenceonAutomation,QualityandTesting,Robotics(AQTR)*,2020,pp.1–6.DOI:10.1109/AQTR49680.2020.9129886.

3.]M.A.Tamiru,J.Tordsson,E.Elmroth,andG.Pierre,"Anexperimentalevaluation ofthekubernetesclusterautoscalerinthecloud,"in*2020IEEEInternationalConferenceonCloudComputingTechnologyandScience(CloudCom)*,2020,pp.17–24. DOI:10.1109/CloudCom49646.2020.00002.V.M. Patro, M.R. Patra, Augmenting Weighted Average with Confusion matrix to Enhance classification accuracy. Trans. Mach,. Learn. Artif Intell. 2(4),77-91(2019)

4. M. A. Rodriguez and R. Buyya, "Containers orchestration with cost-efficient autoscaling in cloud computing environments," *ArXiv*, vol. abs/1812.00300, 2018.

5. J.Watada,A.Roy,R.Kadikar,H.Pham,andB.Xu,"Emergingtrends,techniques andopenissuesofcontainerization:Areview,"*IEEEAccess*,vol.7,pp.152443–152472,2019.DOI:10.1109/ACCESS.2019.2945930.

6. E.Casalicchio,"Astudyonperformancemeasuresforauto-scalingcpu-intensive containerizedapplications,"*ClusterComputing*,vol.22,Sep.2019.DOI:10.1007/s10586-018-02890-1.

7. A.PereiraFerreiraandR.Sinnott,"Aperformanceevaluationofcontainersrunning onmanagedkubernetesservices,"in*2019IEEEInternationalConferenceonCloud ComputingTechnologyandScience(CloudCom)*,2019,pp.199–208.DOI:10.1109/CloudCom.2019.00038.

8. I.M.A.Jawarneh,P.Bellavista,F.Bosi,L.Foschini,G.Martuscelli,R.Montanari,andA.Palopoli,"Containerorchestrationengines:Athoroughfunctionaland performancecomparison,"in*ICC2019-2019IEEEInternationalConferenceon Communications(ICC)*,2019,pp.1–6.DOI:10.1109/ICC.2019.8762053.